Thinking in ClojureScript

Programming is not about typing, it's about thinking - Rich Hickey

What is ClojureScript?

ClojureScript is a compiler for Clojure that targets JavaScript.

Goals

- Understand the ClojureScript thought process
- Learn enough ClojureScript
- Convince YOU to take ClojureScript for a spin
- Take some of these principles back to JavaScript

Why ClojureScript?

- Philosophy
- It's not JavaScript
- Data Structures / Immutability
- core.async library

It's not JavaScript


wat

- Thought out design (`Nan == Nan ;false`)
- Battle tested (i.e. Google Closure)
- Browser Repl
- Realtime feedback
- Macros, namespaces

Figwheel / Browser Repl

Just enough ClojureScript

Hello ClojureScript

```
Hello ClojureScript
;; Hello ClojureScript
```

Function Call

```
(+ 5 (*  5 2))
;; 15

// JavaScript
5 + 5 * 2
```

Defining Functions

```
(defn multiply
  [x, y]
    (* x y))

// JavaScript
function multiply(x, y) {
  x * y
}
```

Interop

```
(. js/document (getElementById "app")
;; method call
(.-value input)
;; property
```

atom

```
(def count (atom 0))

@count
;; 0

(swap! count 1)
(reset! count 1)
```

let

```
(let [x 1]
  x)
```

Macros

```
(-> (om/get-node owner "new-contact-name")
    .-value)
```

```clojure
(defmacro unless
  [pred & body]
  `(if (not ~pred)
     (do ~@body)
     nil))

;; Compiles to
(macroexpand '(unless true (/ 1 0)))
; => (if (clojure.core/not true) (do (/ 1 0)) nil)
```

core.asyc

Clojure's implementation of Communicating Sequential Processes

Communicating Sequential Processes

- processes
- channels
- coordination

Escape callback hell

```
1
2  var async = require("async");
3
4  User.find(userId, function(err, user){
5    if (err) return errorHandler(err);
6    User.all({where: {id: {$in: user.friends}}}, function(err, friends) {
7      if (err) return errorHandler(err);
8      async.each(friends, function(friend, done){
9        friend.posts = [];
10       Post.all({where: {userId: {$in: friend.id}}}, function(err, posts) {
11         if (err) return errorHandler(err);
12         async.each(posts, function(post, donePosts){
13           friend.push(post);
14           Comments.all({where: post.id}, function(err, comments) {
15             if (err) donePosts(err);
16             post.comments = comments;
17             donePosts();
18           });
19         }, function(err) {
20           if (err) return errorHandler(err);
21           done();
22         });
23       });
24     }, function(err) {
25       if (err) return errorHandler(err);
26       render(user, friends);
27     });
28   }
29 });
30
```

Write sequential logic

```
(println "do something")
(send-to-channel)
(println "continue")
```

Primitives

Chan (get better definitions for these)

```
(chan)
```

Put

```
(put!)
(>!)
```

Take

```
(take!)
(<!)
```

go

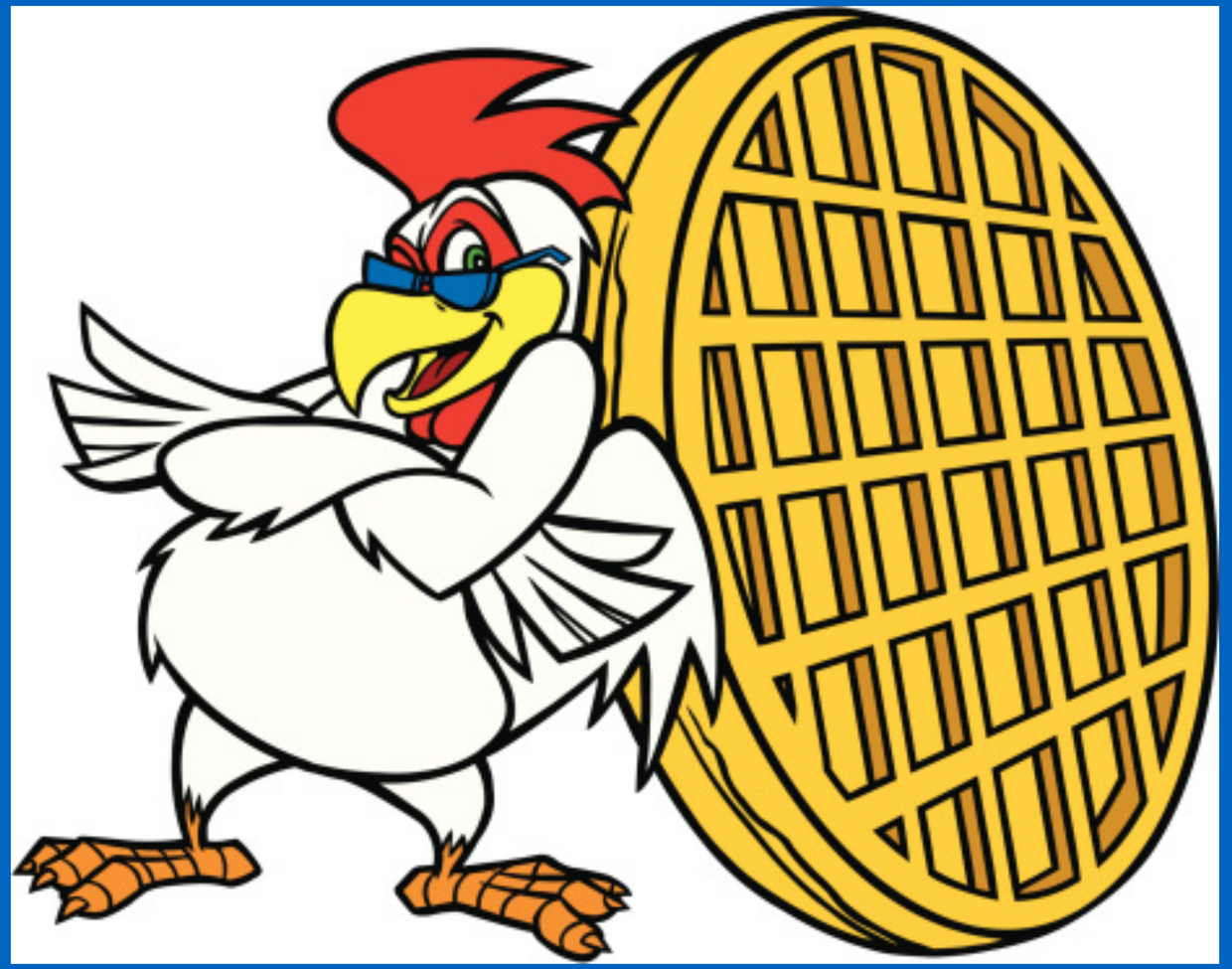
```
(go
  (println "Waiting...")
  (<! events)
  (show! "Got an event")
)
```

Code example

Immutability

```
(def a [1 2 3])
(println (conj a 42))
;; [1 2 3 42]
(println a)
;; [1 2 3]
```

Om + React



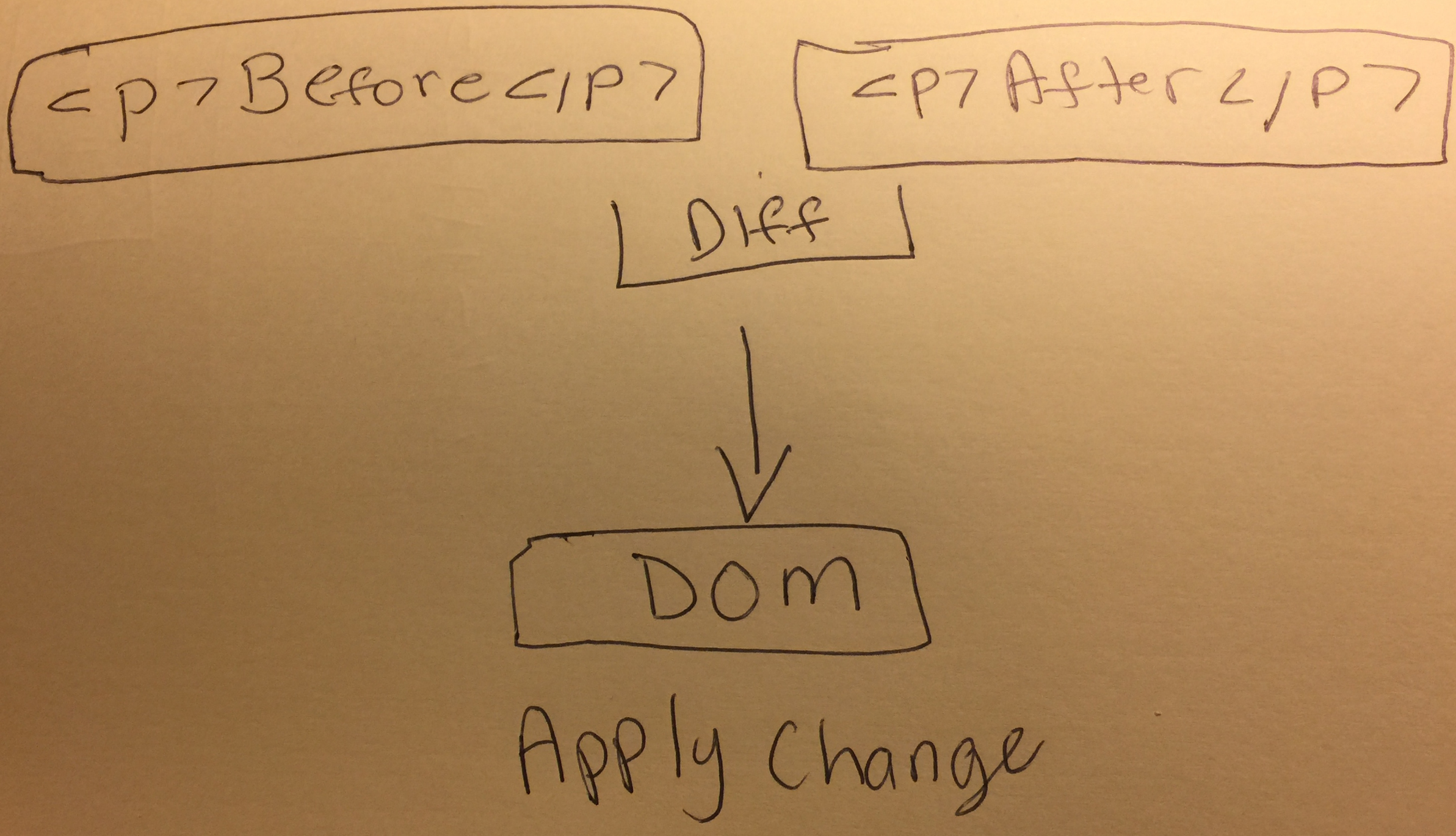
Lets talk about Om

Om

Interface to Facebook's React

React

- V in MVC
- Immediate mode rendering
- Components


Before
After
Diff
DOM
Apply Change

Example

https://github.com/iamjarvo/pplz

Helpful links

- http://funcool.github.io/clojurescript-unraveled/
- https://github.com/circleci/frontend
- https://www.youtube.com/user/ClojureTV
- https://github.com/omcljs/om
- http://swannodette.github.io/

Me

- Jearvon Dharrie
- Twitter: @jearvon
- Podcast: http://turing.cool